

1 **CLAIMS**

2 I claim:

3
4 1. A method, comprising:

5 forming a dictionary containing base entries representing individual
6 instructions in a program and sequence entries representing corresponding
7 sequences of multiple instructions in the program; and
8 generating items that represent the program in terms of the base entries and
9 the sequence entries.

10
11 2. A method as recited in claim 1, wherein the forming comprises
12 creating a split-stream dictionary.

13
14 3. A method as recited in claim 1, wherein the sequence entries
15 represent short sequences consisting of two to four instructions.

16
17 4. A method as recited in claim 1, wherein the sequence entries
18 represent sequences of multiple instructions that are used multiple times in the
19 program.

20
21 5. A method as recited in claim 1, wherein the generating comprises:
22 comparing an input string of instructions to the sequence entries in the
23 dictionary; and
24 if the input string matches a particular sequence entry, generating an item
25 that references the particular sequence entry.

1
2 6. A method as recited in claim 1, wherein the generating comprises:
3 comparing progressively smaller strings of multiple instructions, where
4 each string begins with a first instruction, to the sequence entries in the dictionary;
5 if any string of multiple instructions matches a particular sequence entry,
6 generating a first item that references the particular sequence entry; and
7 if no string of multiple instructions matches the sequence entries,
8 generating a second item that references a base entry associated with the first
9 instruction.

10
11 7. A method as recited in claim 1, further comprising compressing the
12 dictionary.

13
14 8. A method as recited in claim 1, further comprising compressing the
15 base entries of the dictionary.

16
17 9. A method as recited in claim 8, wherein the compressing comprises:
18 sorting the base entries by opcodes to create instruction groups so that there
19 is one instruction group for each opcode; and
20 for each instruction group, sorting the base entries according to size of
21 individual instruction fields and outputting each instruction field as a separate
22 stream.

1 **10.** A method as recited in claim 1, further comprising compressing the
2 sequence entries of the dictionary.
3

4 **11.** A method as recited in claim 10, wherein the compressing
5 comprises constructing tree structures for individual sequences of multiple
6 instructions.
7

8 **12.** A computer readable medium storing the dictionary and the items
9 produced as a result of the method as recited in claim 1.
10

11 **13.** A computer readable medium having computer-executable
12 instructions that, when executed on one or more processors, performs the method
13 as recited in claim 1.
14

15 **14.** A method, comprising:
16 analyzing a program containing multiple instructions;
17 creating base entries in a dictionary for individual instructions; and
18 creating sequence entries in the dictionary for corresponding sequences of
19 multiple instructions that are used multiple times in the program.
20

21 **15.** A method as recited in claim 14, wherein the sequence entries
22 represent short sequences consisting of two to four instructions.
23
24
25

1 **16.** A method as recited in claim 14, further comprising compressing the
2 base entries of the dictionary.

3
4 **17.** A method as recited in claim 16, wherein the compressing
5 comprises:

6 sorting the base entries by opcodes to create instruction groups so that there
7 is one instruction group for each opcode; and

8 for each instruction group, sorting the base entries according to size of
9 individual instruction fields and outputting each instruction field as a separate
10 stream.

11
12 **18.** A method as recited in claim 14, further comprising compressing the
13 sequence entries of the dictionary.

14
15 **19.** A method as recited in claim 18, wherein the compressing
16 comprises constructing tree structures for individual sequences of multiple
17 instructions.

18
19 **20.** A method as recited in claim 14, further comprising generating
20 items that represent the program in terms of the base entries and the sequence
21 entries.

22
23 **21.** A method as recited in claim 20, wherein the generating comprises:
24 comparing progressively smaller strings of multiple instructions, where
25 each string begins with a first instruction, to the sequence entries in the dictionary;

1 if any string of multiple instructions matches a particular sequence entry,
2 generating a first item that references the particular sequence entry; and

3 if no string of multiple instructions matches the sequence entries,
4 generating a second item that references a base entry associated with the first
5 instruction.

6
7 **22.** A computer readable medium storing the dictionary produced as a
8 result of the method as recited in claim 14.

9
10 **23.** A computer readable medium having computer-executable
11 instructions that, when executed on one or more processors, performs the method
12 as recited in claim 14.

13
14 **24.** A method, comprising:
15 creating base entries in a dictionary for individual instructions in a
16 program;
17 creating sequence entries in the dictionary for corresponding sequences of
18 multiple instructions that are used multiple times in the program;
19 compressing the base entries and the sequence entries to produce a
20 compressed dictionary; and
21 generating items that represent the program in terms of the base entries and
22 the sequence entries.

1 **25.** A method as recited in claim 24, wherein the sequence entries
2 represent short sequences consisting of two to four instructions.

3
4 **26.** A method as recited in claim 24, wherein the compressing
5 comprises:

6 sorting the base entries by opcodes to create instruction groups so that there
7 is one instruction group for each opcode; and

8 for each instruction group, sorting the base entries according to size of
9 individual instruction fields and outputting each instruction field as a separate
10 stream.

11
12 **27.** A method as recited in claim 24, wherein the compressing
13 comprises constructing tree structures for individual sequences of multiple
14 instructions.

15
16 **28.** A method as recited in claim 24, wherein the generating comprises:
17 comparing progressively smaller strings of multiple instructions, where
18 each string begins with a first instruction, to the sequence entries in the dictionary;
19 if any string of multiple instructions matches a particular sequence entry,
20 generating a first item that references the particular sequence entry; and

21 if no string of multiple instructions matches the sequence entries,
22 generating a second item that references a base entry associated with the first
23 instruction.

1 **29.** A method as recited in claim 24, further comprising decompressing
2 the compressed dictionary.

3
4 **30.** A method as recited in claim 29, further comprising translating the
5 items back to the instructions by using the base entries and the sequence entries of
6 the dictionary.

7
8 **31.** A computer readable medium having computer-executable
9 instructions that, when executed on one or more processors, performs the method
10 as recited in claim 24.

11
12 **32.** A method for decoding a file derived from a program, the file
13 having a dictionary with base entries representing individual instructions in the
14 program and sequence entries representing corresponding sequences of multiple
15 instructions in the program and multiple items that represent the program in terms
16 of the base entries and the sequence entries, the method comprising:

17 recovering the base entries and the sequence entries of the dictionary; and
18 translating the items to instructions in the program by using the base entries
19 and the sequence entries in the dictionary.

20
21 **33.** A method as recited in claim 32, wherein the dictionary is
22 compressed and the recovering comprises decompressing the compressed
23 dictionary.

1 **34.** A method as recited in claim 32, wherein the translating comprises
2 copying the base entries and the sequence entries into a code buffer.

3
4 **35.** A computer readable medium having computer-executable
5 instructions that, when executed on one or more processors, performs the method
6 as recited in claim 32.

7
8 **36.** A computer readable medium having computer-executable
9 instructions that, when executed on one or more processors, directs a computing
10 device to:

11 read a program containing multiple instructions;
12 create base entries in a dictionary for individual instructions in the program;
13 create sequence entries in the dictionary for corresponding sequences of
14 multiple instructions that are used multiple times in the program; and
15 generate items that represent the program in terms of the base entries and
16 the sequence entries

17
18 **37.** A computer readable medium as recited in claim 36, wherein the
19 sequence entries represent short sequences consisting of two to four instructions.

20
21 **38.** A computer readable medium as recited in claim 36, further
22 comprising instructions to compress the dictionary.

1 **39.** A computer readable medium as recited in claim 36, further
2 comprising instructions to:

3 sort the base entries by opcodes to create instruction groups so that there is
4 one instruction group for each opcode; and

5 for each instruction group, sort the base entries according to size of
6 individual instruction fields and outputting each instruction field as a separate
7 stream.

8
9 **40.** A computer readable medium as recited in claim 36, further
10 comprising instructions to compress the sequence entries by constructing tree
11 structures for individual sequences of multiple instructions.

12
13 **41.** A computer readable medium as recited in claim 36, further
14 comprising instructions to:

15 compare progressively smaller strings of multiple instructions, where each
16 string begins with a first instruction, to the sequence entries in the dictionary;

17 if any string of multiple instructions matches a particular sequence entry,
18 generate a first item that references the particular sequence entry; and

19 if no string of multiple instructions matches the sequence entries, generate a
20 second item that references a base entry associated with the first instruction.

21
22 **42.** A program compression architecture comprising:

23 a dictionary builder to construct a dictionary containing base entries
24 representing individual instructions in a program and sequence entries
25

1 representing corresponding sequences of multiple instructions that are used
2 multiple times in the program; and

3 an item generator to generate items that represent the program in terms of
4 the base entries and the sequence entries.

5
6 **43.** A program compression architecture as recited in claim 42 wherein
7 the sequence entries represent short sequences consisting of two to four
8 instructions.

9
10 **44.** A program compression architecture as recited in claim 42 wherein
11 the item generator is configured to compare an input string of instructions to the
12 sequence entries in the dictionary and if the input string matches a particular
13 sequence entry, generate an item that references the particular sequence entry.

14
15 **45.** A program compression architecture as recited in claim 42 wherein
16 the item generator is configured to compare progressively smaller strings of
17 multiple instructions, where each string begins with a first instruction, to the
18 sequence entries in the dictionary such that (1) if any string of multiple
19 instructions matches a particular sequence entry, the item generator produces a
20 first item that references the particular sequence entry and (2) if no string of
21 multiple instructions matches the sequence entries, the item generator produces a
22 second item that references a base entry associated with the first instruction.

1 **46.** A program compression architecture as recited in claim 42 further
2 comprising a dictionary compressor to compress the dictionary.

3
4 **47.** A program compression architecture as recited in claim 46 wherein
5 the dictionary compressor is configured to compress the base entries
6 independently of the sequence entries.

7
8 **48.** A program compression architecture as recited in claim 46 wherein
9 the dictionary compressor is configured to sort the base entries by opcodes to
10 create instruction groups so that there is one instruction group for each opcode, the
11 dictionary compressor being further configured to sort the base entries within each
12 instruction group according to size of individual instruction fields and output each
13 instruction field as a separate stream.

14
15 **49.** A program compression architecture as recited in claim 46 wherein
16 the dictionary compressor is configured to construct tree structures for individual
17 sequences of multiple instructions.

18
19 **50.** An embedded system comprising the program compression
20 architecture of claim 42.

21
22 **51.** A computer comprising:
23 a memory;
24 a processing unit coupled to the memory; and
25

1 a program compression system stored in the memory and executable on the
2 processing unit, the program compression system building a dictionary containing
3 base entries representing individual instructions in a program and sequence entries
4 representing corresponding sequences of multiple instructions in the program, the
5 program compression system generating items that represent the program in terms
6 of the base entries and the sequence entries.

7
8 **52.** A computer as recited in claim 51, wherein the program
9 compression system is further configured to compress the dictionary.

10
11 **53.** A data structure stored on a computer readable medium, comprising:
12 base entries representing individual instructions in a program; and
13 sequence entries representing corresponding sequences of multiple
14 instructions that are used multiple times in the program, the sequence entries
15 referencing the base entries.

16
17 **54.** A data structure stored as recited in claim 53, further comprising
18 items that reference the base entries and the sequence entries to represent
19 instruction strings in the program.
20
21
22
23
24
25